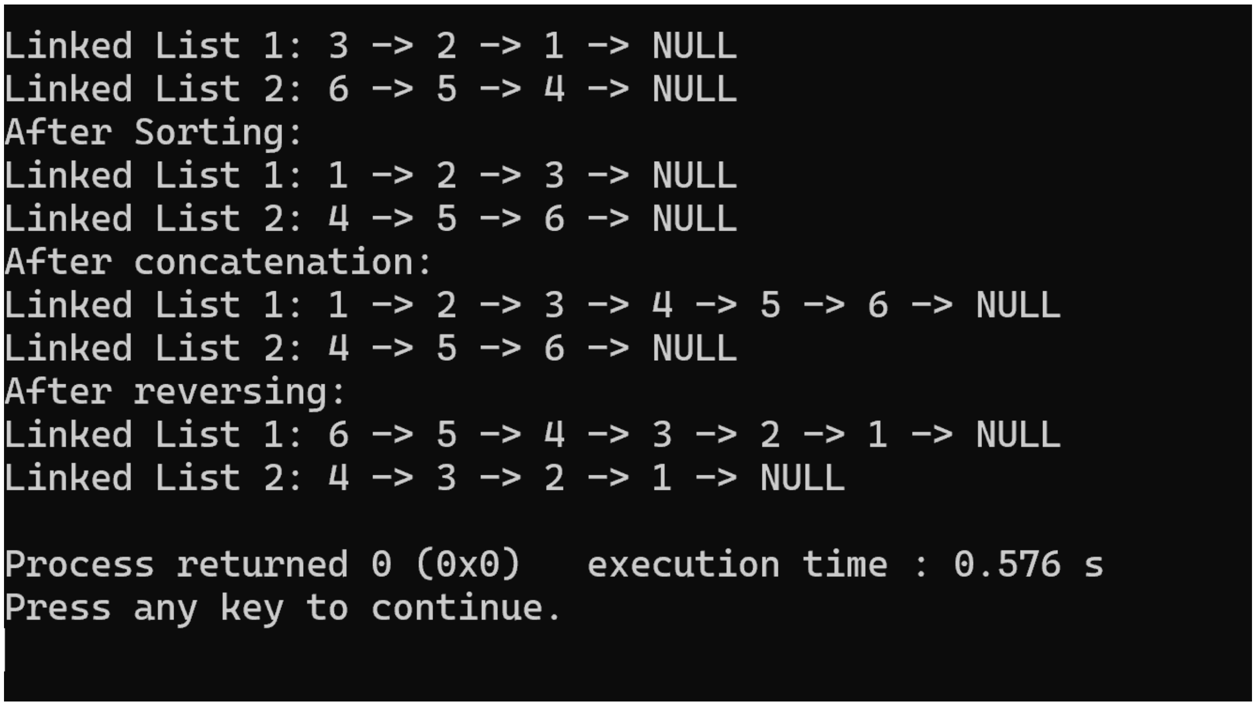
```c
#include <stdio.h>

#include <stdlib.h>

typedef struct Node

{

int data;

struct Node *next;

} Node;

struct Node *createNode(int value)

{

struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));

if (newNode == NULL)

{

printf("Memory allocation failed.\n");

exit(1);

}

newNode->data = value;

newNode->next = NULL;

return newNode;

}

struct Node *insertAtBeginning(struct Node *head, int value)

{


struct Node *newNode = createNode(value);

newNode->next = head;

return newNode;

}

struct Node *concat(Node *head1, Node *head2)

{

Node *temp = head1;
while (temp->next != NULL)

temp = temp->next;

temp->next = head2;

return head1;

}

struct Node *sort(Node *head)

{

Node *temp, *current;

int t;

current = head;

while (current != NULL)

{

temp = head;

while (temp->next != NULL)


{

if (temp->data > temp->next->data)

{

t = temp->data;

temp->data = temp->next->data;

temp->next->data = t;

}

temp = temp->next;

}

current = current->next;

}

return head;

}

struct Node *reverse(Node *head)
```

```
{
Node *prev, *temp, *next;
temp = head;
prev = NULL;
while (temp != NULL)
{
next = temp->next;

temp->next = prev;
prev = temp;
temp = next;
}
head = prev;
return head;
}
void displayLinkedLists(struct Node *head1, struct Node *head2)
{
printf("Linked List 1: ");
while (head1 != NULL)
{
printf("%d -> ", head1->data);
head1 = head1->next;
}
printf("NULL\n");
printf("Linked List 2: ");
while (head2 != NULL)
{
printf("%d -> ", head2->data);
head2 = head2->next;

}

printf("NULL\n");

}

int main()

{

struct Node *list1 = NULL;

struct Node *list2 = NULL;

int choice, data;

list1 = insertAtBeginning(list1, 1);

list1 = insertAtBeginning(list1, 2);

list1 = insertAtBeginning(list1, 3);

list2 = insertAtBeginning(list2, 4);

list2 = insertAtBeginning(list2, 5);

list2 = insertAtBeginning(list2, 6);

displayLinkedLists(list1, list2);

printf("After Sorting:\n");

list1 = sort(list1);

list2 = sort(list2);

displayLinkedLists(list1, list2);

printf("After concatenation:\n");

list1 = concat(list1, list2);

displayLinkedLists(list1, list2);


printf("After reversing:\n");

list1 = reverse(list1);

displayLinkedLists(list1, list2);

return 0;

}
```



```
Linked List 1: 3 -> 2 -> 1 -> NULL
Linked List 2: 6 -> 5 -> 4 -> NULL
After Sorting:
Linked List 1: 1 -> 2 -> 3 -> NULL
Linked List 2: 4 -> 5 -> 6 -> NULL
After concatenation:
Linked List 1: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> NULL
Linked List 2: 4 -> 5 -> 6 -> NULL
After reversing:
Linked List 1: 6 -> 5 -> 4 -> 3 -> 2 -> 1 -> NULL
Linked List 2: 4 -> 3 -> 2 -> 1 -> NULL

Process returned 0 (0x0)   execution time : 0.576 s
Press any key to continue.
```